Documentation of Logic Used:

1. Heuristic Repair/Stochastic Search:

This algorithm follows a hill-climbing approach. It begins by placing queens randomly using the NQueens.place_n_queens() function and then iteratively adjusts their positions to reduce conflicts. In each iteration, the queen is moved to the position that results in the fewest conflicts, and this process continues until no conflicts remain. Conflicts are determined by queens occupying the same row, column, or diagonal. If the algorithm gets stuck (i.e., the conflict count remains unchanged for 100 iterations), the board is reset with a new random configuration. The solution has been tested for board sizes between n = 10 and n = 30.

2. Forward Tracking:

This forward-checking algorithm builds upon the structure of the traditional DFS/backtracking algorithm but optimizes performance using a stack (stack) and a dictionary (open_spots). The stack holds previous board states by copying them, allowing the algorithm to backtrack efficiently when necessary. The open_spots dictionary tracks available positions for each row, enabling the algorithm to quickly check whether the next row has any valid positions before attempting to place a queen. By preemptively verifying available spots and using the stack to manage state transitions, the algorithm drastically reduces the number of moves, iterations, and overall time compared to the standard DFS/backtracking approach.